EXHIBIT A

Published on KernelTrap (http://kerneltrap.org)

Interview: William Lee Irwin III

By
Created Mar 12 2002 - 16:14

William Lee Irwin III, AKA "wli" on the #kernelnewbies IRC channel is one of the developers helping to implement a reverse mapping feature into the Linux kernel.

Randy Hron: How long have you been working with operating systems? Unix? Linux?

William Lee Irwin III: It depends on what you consider working with an operating system. Systems programming, i.e. writing code that runs in the kernel, I've only been doing since around the time I got hired by IBM.

I've used Linux as a userland programming environment since 1996, and my first computing experience was on the Purdue ECN lab systems in 1993, which consisted of a number of Visual Graphics X-19 terminals and Sun IPC's used as terminals for a Sequent S-81. Before 2000 I had no involvement whatsoever with kernel programming. By and large Solaris was my primary programming environment before 2000 (of course back then I had nothing to do with its or any kernel), though there was a little IRIX in there, too.

rwh: How long have you been with IBM? How has Linux' position with IBM changed over time?

wli: I've worked for IBM since April of 2000. When I started, I worked for Sequent, which was being absorbed into IBM. IBM is a very large company, so at different sites, there are different areas of specialty and focus. I can't pretend to be very knowledgable regarding the scope or history of IBM's business plans, but at least for me, the progression was from DYNIX/ptx to AIX and then to Linux, while learning progressively more about systems programming.

For a while, there was not much focus on-site for Linux, but at some point a decision was made that that would be our site's focus, and this seems to be working out well.

rwh: How is Linux different than other projects you've been involved with?

wli: Well, Linux has a much broader userbase and also much broader visibility. By and large the issues are similar. UNIX kernels are large programs everywhere, and the programming involved is both low-level and complex everywhere. Linux is no exception. Some other issues are largely superficial from the standpoint of how it feels to participate in a project. For instance, things like the raw numbers of people and lines of code don't seem to make much difference beyond orders of magnitude. Things like coding style, while they're very important, are also superficial: each project has a style standard, and you follow it. So as far as operating systems go, the distinguishing

features of Linux don't really seem to have much impact on what it's like to program for it.

There is probably one major difference that seem to come up, but it's not something that really affect day-to-day operation. It's that Linux is a competitive environment. Many other efforts don't seem to have multiple people presenting competing solutions for the same issue, yet in Linux this is common. The primary difference this appears to make is that people who disagree about how something should be implementated have working code to show that could displace yours.

rwh: What do you enjoy most about programming/mathematics?

wli: It's odd to admit it but I think that I've actually discovered more about the kind of programming I like to do and the kind of mathematics I enjoy doing now that I'm out of school than in it. Now that I have been under my own direction, the kinds of mathematics I've developed interest in have been more calculational, and the kind of programming I've developed interests in have been more side-effect-driven than what I thought I was interested in. Specifically, the mathematics I've become interested in are things like 19th-century material on elliptic functions, and the programming I've become interested in has been (of course) kernel programming. Essentially I've discovered that I actually like doing things with concretely observable results.

I think the key to understanding what I like to do is to go out and do things on whims, and then in retrospect analyze what I've done to discern what I like, and allow myself to naturally gravitate to what I want to do. This is in opposition to doing things with a preconceived notion of whether I'll like it or not, which approach I believe resulted in some missed opportunities for fun and achievement earlier on for me. There are, of course, clear limits to how far this idea should be taken.

Now, if I'm to analyze in retrospect what I like about mathematics and what I like about programming, it's hard to characterize in formal terms. With all apologies to Dijkstra, I'll have to resort to analogy. It's like I am a child in the realm of proofs and programs and syllogisms and hypotheses and lemmas and hash tables and binary search trees are my building blocks, and I play by assembling my tinkertoys into mighty mighty cathedrals, brick by brick, stone by stone. And perhaps one day after rebuilding the world within the machine sufficiently many times I'll get it right and live within a crystal palace, but I suspect that the iteration is not a finite process.

rwh: It sounds like you have a background in computer science. Where did you take your education?

wli: I have a bachelor of science degree from Purdue University, where I majored in mathematics and computer science.

rwh: What do you enjoy doing when you aren't working?

*wli:*Well, the trick is that programming is my main form of entertainment. So from one standpoint I get paid to goof off all day. =)

I do do other things though. There is the usual assortment of nightclubs to visit and concerts to listen to now and then, and I do a number of ordinary things that aren't programming, like watch anime and read fiction and so on. They're fairly distant seconds, but they're there.

There's also IRC, but that's more of a supplement (or maybe even replacement) for newsgroups and mailing lists calling people on the telephone, and a very effective one. Cutting and pasting code in realtime to people halfway around the world is extremely convenient, as is the ability to discuss algorithms and designs and so on.

rwh: You have eclectic musical taste. What music appeals to you most?

wli: I'm not sure what to say here. I'm not a particular expert in music. I do collect some and I collect what I like (of course). A list of the artists and albums would be too banal to bear stating.

It's difficult to characterize the sorts of music I like entirely in terms of genre, but there is a distinct pattern. To me content of music is neither words nor musical notes, but the atmosphere it creates and the emotional response it triggers. Whether it's calculated or a coincidence, technically advanced or crude banging on instruments, the worth of the resulting sounds is judged only by the power of the effect.

There appear to be several distinct themes that emerge from contemplating my own playlists. One is a grandiose hyperaggressive lunge toward empowerment. Another is a mournful grieving cry. And another is escapism through deliberate complexity. And of course, humor.

rwh: Any tips for the aspiring kernel hacker?

wli: Well, I believe I am still aspiring myself. =)

Everyone says persistence is key, but it's not enough. One thing I've noticed is that because the kernel is responsible for maintaining the integrity of both data and the running system image, the cost of a failure (i.e. bugs, and as with any programming, they are numerous) is that data is lost and systems go down. A big fear to overcome is that of disrupting the proper operation of a system or losing data. When a kernel crashes it destroys data and the machine goes down, and you can't be afraid to see this happen if you're going to get anywhere. Programming is error-prone, and one must be prepared to commit errors. A stumbling block for me early on was that I was too careful and obsessed on repeatedly reviewing code to be sure the system wouldn't crash when I tried to run with it.

This is ineffective. A more effective approach appears to be creating a sandbox where the data is disposable and the system nonessential and running the code and figuring out what went wrong when the system crashes. And it's not easy. Without much hardware assistance (requiring too much money to be practical) it's generally not possible to recover much of the system state after the event, so working around this by dumping state at the appropriate times or running within a simulator (fortunately, bochs is free as in beer) is required. Some infrastructures exist, e.g. kgdb and kdb. I'm already espousing perhaps heretical notions, but I don't care. And another thing is that reading code is harder than writing it (and debugging is harder than both but moving on) so a from-scratch rewrite of something will be easier than finding the small changes needed to fix real problems. For regular kernel hacking, rewrites aren't going to get anywhere, those who wrote the originals will scream bloody murder and those who have to call the stuff are terrified they'll have to deal with new bugs in unfamiliar code. But as a crutch for getting around not quite being able to read things it's fine. Maybe someone will come after me for saying so as there are bound to be frivolous rewrites of all kinds of things after any kind of public statement like this, but if people get off their butts and stop duplicating everyone else's merges of $VM + O(1) + misc garbage to write

some actual new code, it's worth the flames.

So I look at the above and it's somewhat wordy for the intent to be clear to everyone. Write something and don't stall yourself with excessive cautiousness, and then rapidly begin testing.

Do something and do something new!

Related Links:

- William Lee Irwin III's Home Page [1] - (http://holomorphy.com/~wli)
- wli's kernel.org repository [2] - (http://www.kernel.org/pub/linux/kernel/people/wli/)

About the interviewer:

Randy Hron is a Unix/Oracle administrator who kicked Solaris off his main home computer when Mandrake 6.0 was released. His home page is http://home.earthlink.net/~rwhron/ [3]

Source URL:
http://kerneltrap.org/node/80

Links:
[1] http://holomorphy.com/~wli
[2] http://www.kernel.org/pub/linux/kernel/people/wli/
[3] http://home.earthlink.net/~rwhron/